LLMs for Databases: the Next Five Years

Immanuel Trummer

How do You Know a Human Wrote This?
The New York Times, 2020

Meet GPT-3. It Has Learned to Code (and Blog and Argue).
The New York Times, 2020

An A.I. bot answers 10 burning questions about the 2020 NFL season
USA Today, 2020

Meet GPT-3, the natural-language system that generates tweets, pens poetry, summarizes emails, answers trivia, translates languages and even writes its own computer programs
Chicago Tribune, 2021

The Jessica Simulation: Love and loss in the age of A.I.
The San Francisco Chronicle, 2021

Bringing People Back to Life With the Power of AI Chatbots.
Forbes, 2021

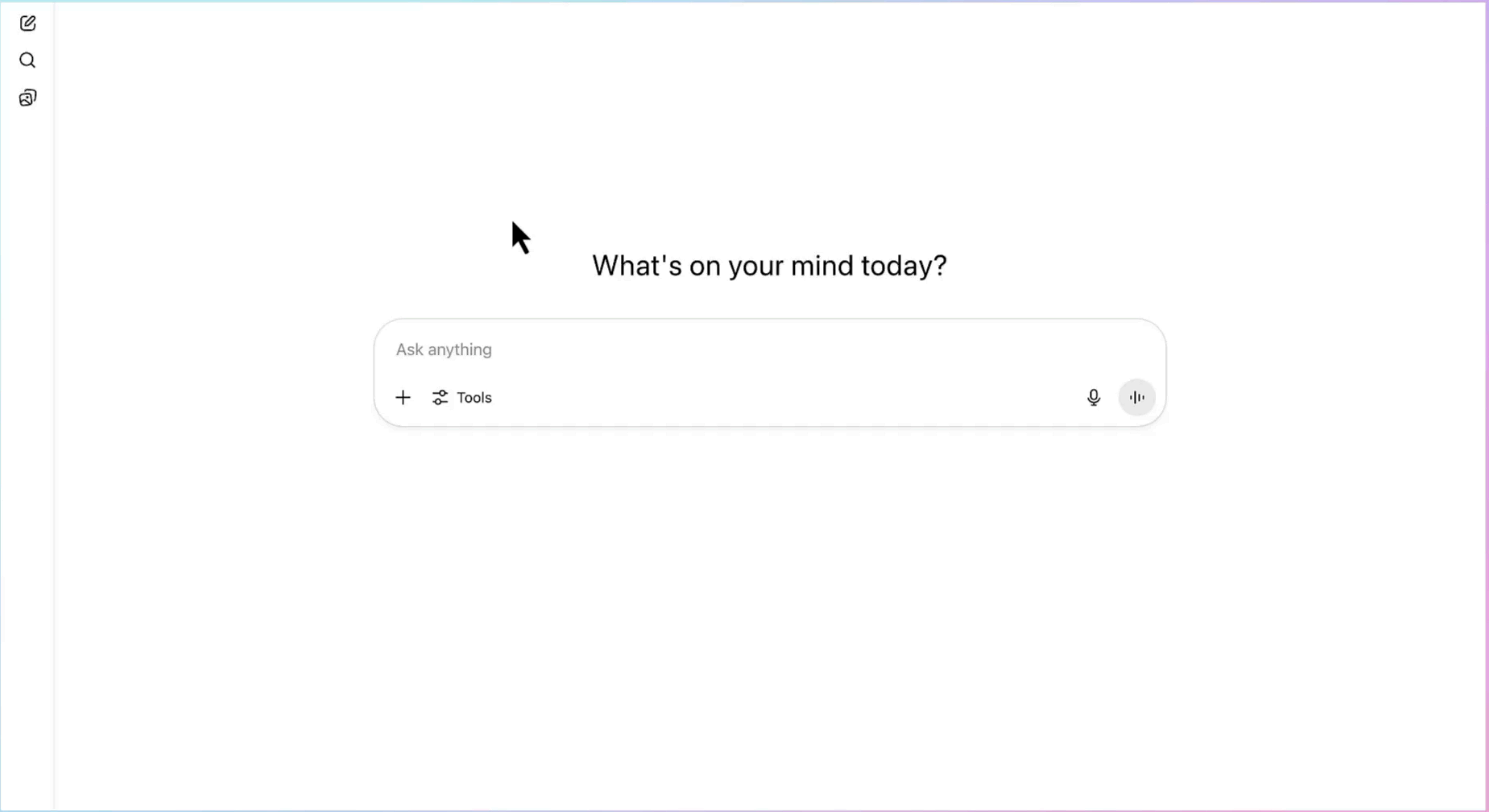
'Grassroots' bot campaigns are coming. Governments don't have a plan to stop them.
Washington Post, 2021

ChatGPT Could be AI's iPhone Moment
Bloomberg, 2022

The New Chatbots Could Change the World.
The New York Times, 2022

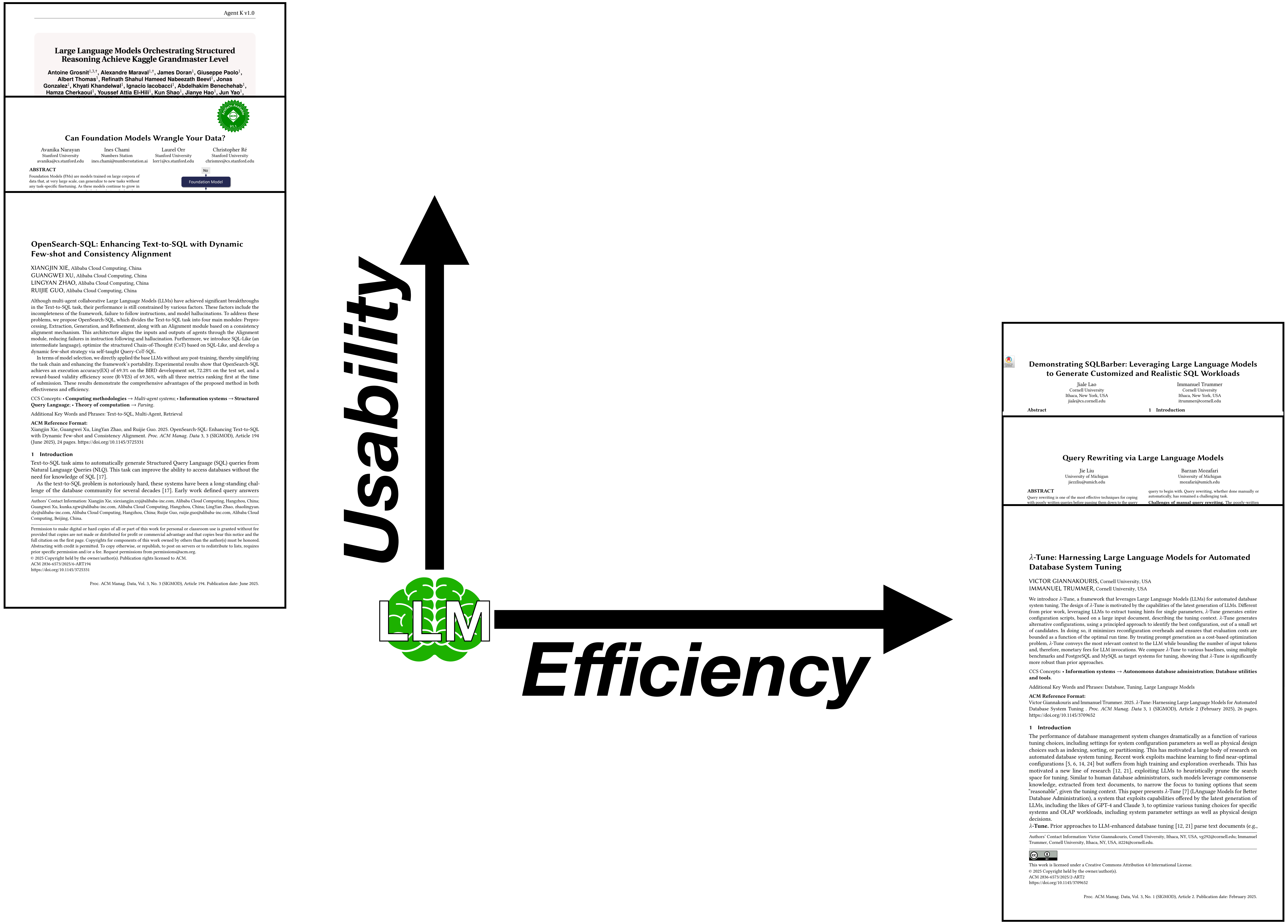
ChatGPT Gained One Million Users in Under a Week. Here's Why the Chatbot is Primed to Disrupt Search as We Know It.
Fortune, 2022

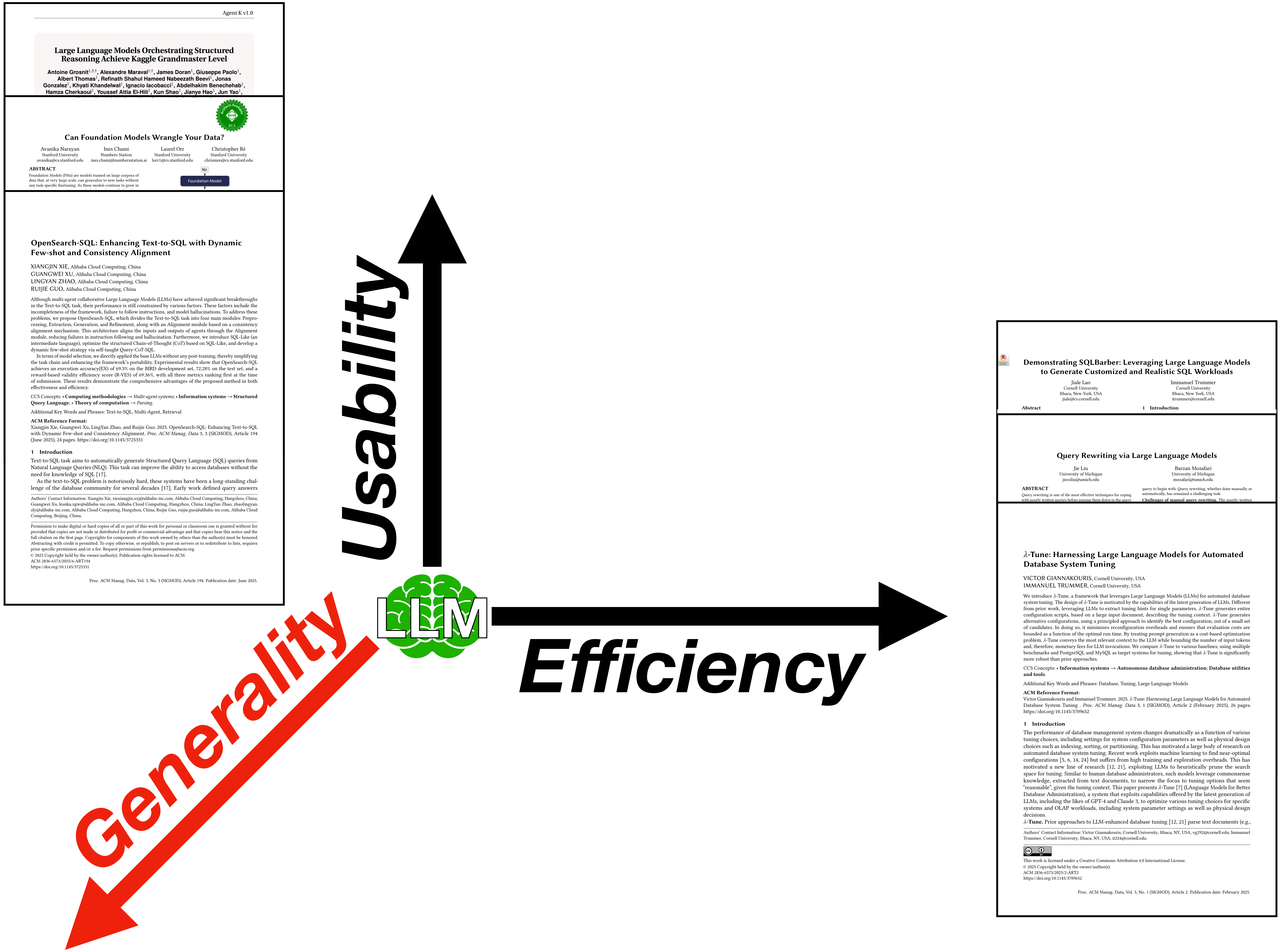
ChatGPT and How AI Disrupts Industries.
Harvard Business Review, 2022

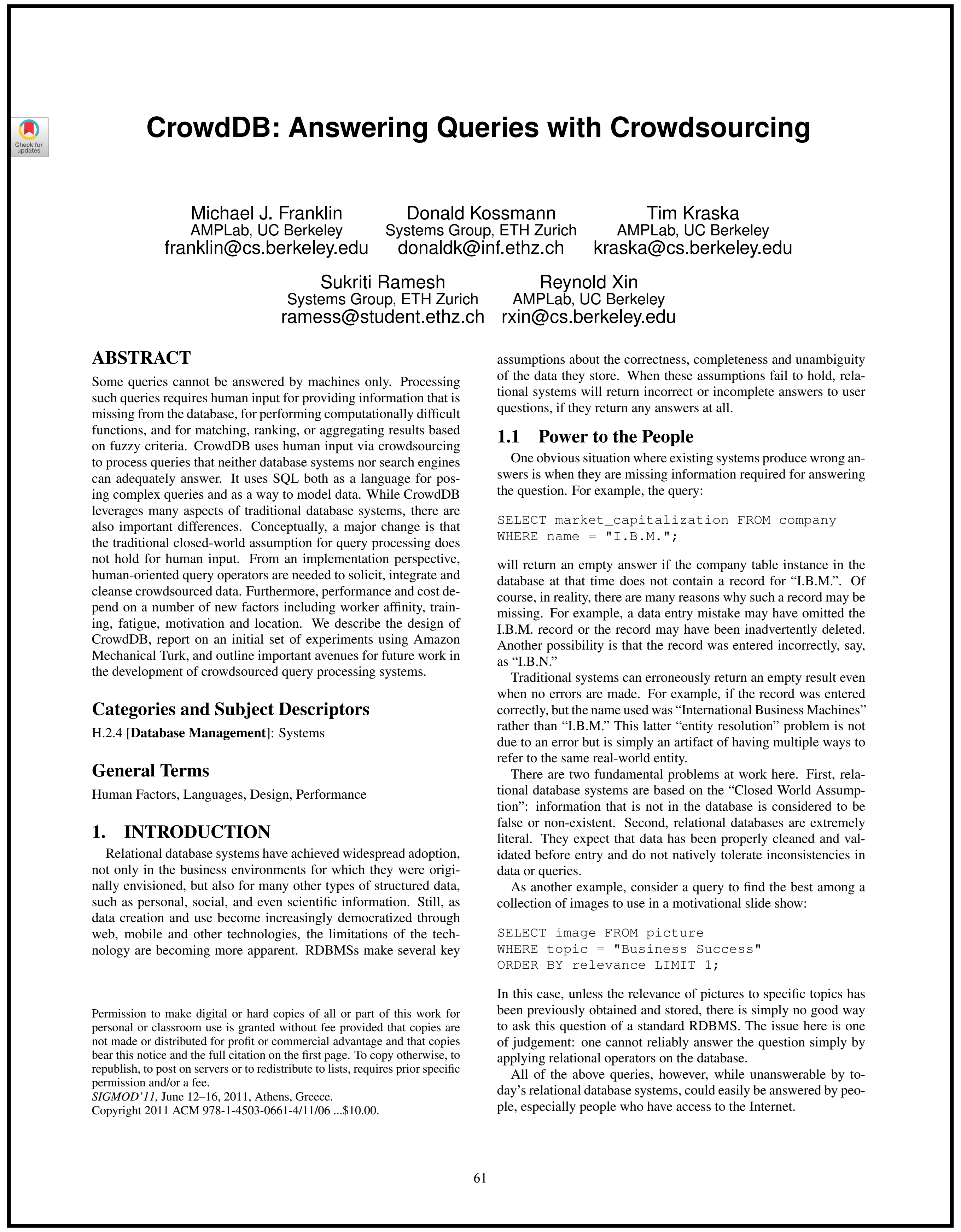

What's on your mind today?
Ask anything
Tools

LLM


Agent K v1.0
Large Language Models Orchestrating Structured Reasoning Achieve Kaggle Grandmaster Level
Antoine Grosnit[1,3,†], Alexandre Maraval[1,†], James Doran[1], Giuseppe Paolo[1], Albert Thomas[1], Refinath Shahul Hameed Nabeezath Beevi[1], Jonas Gonzalez[1], Khyati Khandelwal[1], Ignacio Iacobacci[1], Abdelhakim Benechehab[1], Hamza Cherkaoui[1], Youssef Attia El-Hili[1], Kun Shao[1], Jianye Hao[1], Jun Yao[1],
Artifacts Available
acm
V1.1
Can Foundation Models Wrangle Your Data?
Avanika Narayan
Stanford University
avanika@cs.stanford.edu
Ines Chami
Numbers Station
ines.chami@numbersstation.ai
Laurel Orr
Stanford University
lorr1@cs.stanford.edu
Christopher Ré
Stanford University
chrismre@cs.stanford.edu
ABSTRACT
Foundation Models (FMs) are models trained on large corpora of data that, at very large scale, can generalize to new tasks without any task-specific finetuning. As these models continue to grow in
No
Foundation Model
OpenSearch-SQL: Enhancing Text-to-SQL with Dynamic Few-shot and Consistency Alignment
XIANGJIN XIE, Alibaba Cloud Computing, China
GUANGWEI XU, Alibaba Cloud Computing, China
LINGYAN ZHAO, Alibaba Cloud Computing, China
RUIJIE GUO, Alibaba Cloud Computing, China
Although multi-agent collaborative Large Language Models (LLMs) have achieved significant breakthroughs in the Text-to-SQL task, their performance is still constrained by various factors. These factors include the incompleteness of the framework, failure to follow instructions, and model hallucinations. To address these problems, we propose OpenSearch-SQL, which divides the Text-to-SQL task into four main modules: Preprocessing, Extraction, Generation, and Refinement, along with an Alignment module based on a consistency alignment mechanism. This architecture aligns the inputs and outputs of agents through the Alignment module, reducing failures in instruction following and hallucination. Furthermore, we introduce SQL-Like (an intermediate language), optimize the structured Chain-of-Thought (CoT) based on SQL-Like, and develop a dynamic few-shot strategy via self-taught Query-CoT-SQL.
In terms of model selection, we directly applied the base LLMs without any post-training, thereby simplifying the task chain and enhancing the framework's portability. Experimental results show that OpenSearch-SQL achieves an execution accuracy(EX) of 69.3% on the BIRD development set, 72.28% on the test set, and a reward-based validity efficiency score (R-VES) of 69.36%, with all three metrics ranking first at the time of submission. These results demonstrate the comprehensive advantages of the proposed method in both effectiveness and efficiency.
CCS Concepts: • Computing methodologies → Multi-agent systems; • Information systems → Structured Query Language; • Theory of computation → Parsing.
Additional Key Words and Phrases: Text-to-SQL, Multi-Agent, Retrieval
ACM Reference Format:
Xiangjin Xie, Guangwei Xu, LingYan Zhao, and Ruijie Guo. 2025. OpenSearch-SQL: Enhancing Text-to-SQL with Dynamic Few-shot and Consistency Alignment. Proc. ACM Manag. Data 3, 3 (SIGMOD), Article 194 (June 2025), 24 pages. https://doi.org/10.1145/3725331
1 Introduction
Text-to-SQL task aims to automatically generate Structured Query Language (SQL) queries from Natural Language Queries (NLQ). This task can improve the ability to access databases without the need for knowledge of SQL [17].
As the text-to-SQL problem is notoriously hard, these systems have been a long-standing challenge of the database community for several decades [17]. Early work defined query answers
Authors' Contact Information: Xiangjin Xie, xiexiangjin.xxj@alibaba-inc.com, Alibaba Cloud Computing, Hangzhou, China; Guangwei Xu, kunka.xgw@alibaba-inc.com, Alibaba Cloud Computing, Hangzhou, China; LingYan Zhao, zhaolingyan.zly@alibaba-inc.com, Alibaba Cloud Computing, Hangzhou, China; Ruijie Guo, ruijie.guo@alibaba-inc.com, Alibaba Cloud Computing, Beijing, China.
Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM 2836-6573/2025/6-ART194
https://doi.org/10.1145/3725331
Proc. ACM Manag. Data, Vol. 3, No. 3 (SIGMOD), Article 194. Publication date: June 2025.
Usability
LLM


Agent K v1.0
Large Language Models Orchestrating Structured Reasoning Achieve Kaggle Grandmaster Level
Can Foundation Models Wrangle Your Data?
OpenSearch-SQL: Enhancing Text-to-SQL with Dynamic Few-shot and Consistency Alignment
Usability
LLM
Efficiency
Demonstrating SQLBarber: Leveraging Large Language Models to Generate Customized and Realistic SQL Workloads
Query Rewriting via Large Language Models
λ-Tune: Harnessing Large Language Models for Automated Database System Tuning


Usability
Efficiency
Generality
LLM
Large Language Models Orchestrating Structured Reasoning Achieve Kaggle Grandmaster Level
Can Foundation Models Wrangle Your Data?
OpenSearch-SQL: Enhancing Text-to-SQL with Dynamic Few-shot and Consistency Alignment
Demonstrating SQLBarber: Leveraging Large Language Models to Generate Customized and Realistic SQL Workloads
Query Rewriting via Large Language Models
λ-Tune: Harnessing Large Language Models for Automated Database System Tuning



CrowdDB: Answering Queries with Crowdsourcing

Michael J. Franklin
AMPLab, UC Berkeley
franklin@cs.berkeley.edu

Donald Kossmann
Systems Group, ETH Zurich
donaldk@inf.ethz.ch

Tim Kraska
AMPLab, UC Berkeley
kraska@cs.berkeley.edu

Sukriti Ramesh
Systems Group, ETH Zurich
ramess@student.ethz.ch

Reynold Xin
AMPLab, UC Berkeley
rxin@cs.berkeley.edu

ABSTRACT

Some queries cannot be answered by machines only. Processing such queries requires human input for providing information that is missing from the database, for performing computationally difficult functions, and for matching, ranking, or aggregating results based on fuzzy criteria. CrowdDB uses human input via crowdsourcing to process queries that neither database systems nor search engines can adequately answer. It uses SQL both as a language for posing complex queries and as a way to model data. While CrowdDB leverages many aspects of traditional database systems, there are also important differences. Conceptually, a major change is that the traditional closed-world assumption for query processing does not hold for human input. From an implementation perspective, human-oriented query operators are needed to solicit, integrate and cleanse crowdsourced data. Furthermore, performance and cost depend on a number of new factors including worker affinity, training, fatigue, motivation and location. We describe the design of CrowdDB, report on an initial set of experiments using Amazon Mechanical Turk, and outline important avenues for future work in the development of crowdsourced query processing systems.

Categories and Subject Descriptors

H.2.4 [**Database Management**]: Systems

General Terms

Human Factors, Languages, Design, Performance

1. INTRODUCTION

Relational database systems have achieved widespread adoption, not only in the business environments for which they were originally envisioned, but also for many other types of structured data, such as personal, social, and even scientific information. Still, as data creation and use become increasingly democratized through web, mobile and other technologies, the limitations of the technology are becoming more apparent. RDBMSs make several key assumptions about the correctness, completeness and unambiguity of the data they store. When these assumptions fail to hold, relational systems will return incorrect or incomplete answers to user questions, if they return any answers at all.

1.1 Power to the People

One obvious situation where existing systems produce wrong answers is when they are missing information required for answering the question. For example, the query:

```
SELECT market_capitalization FROM company
WHERE name = "I.B.M.";
```

will return an empty answer if the company table instance in the database at that time does not contain a record for "I.B.M.". Of course, in reality, there are many reasons why such a record may be missing. For example, a data entry mistake may have omitted the I.B.M. record or the record may have been inadvertently deleted. Another possibility is that the record was entered incorrectly, say, as "I.B.N."

Traditional systems can erroneously return an empty result even when no errors are made. For example, if the record was entered correctly, but the name used was "International Business Machines" rather than "I.B.M." This latter "entity resolution" problem is not due to an error but is simply an artifact of having multiple ways to refer to the same real-world entity.

There are two fundamental problems at work here. First, relational database systems are based on the "Closed World Assumption": information that is not in the database is considered to be false or non-existent. Second, relational databases are extremely literal. They expect that data has been properly cleaned and validated before entry and do not natively tolerate inconsistencies in data or queries.

As another example, consider a query to find the best among a collection of images to use in a motivational slide show:

```
SELECT image FROM picture
WHERE topic = "Business Success"
ORDER BY relevance LIMIT 1;
```

In this case, unless the relevance of pictures to specific topics has been previously obtained and stored, there is simply no good way to ask this question of a standard RDBMS. The issue here is one of judgement: one cannot reliably answer the question simply by applying relational operators on the database.

All of the above queries, however, while unanswerable by today's relational database systems, could easily be answered by people, especially people who have access to the Internet.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
SIGMOD'11, June 12–16, 2011, Athens, Greece.
Copyright 2011 ACM 978-1-4503-0661-4/11/06 ...$10.00.

61

Crowdsourced Databases: Query Processing with People

Adam Marcus, Eugene Wu, David R. Karger, Samuel Madden, Robert C. Miller
MIT CSAIL
{marcua, sirrice, karger, madden, rcm}@csail.mit.edu

ABSTRACT

Amazon's Mechanical Turk ("MTurk") service allows users to post short tasks ("HITs") that other users can receive a small amount of money for completing. Common tasks on the system include labelling a collection of images, combining two sets of images to identify people which appear in both, or extracting sentiment from a corpus of text snippets. Designing a workflow of various kinds of HITs for filtering, aggregating, sorting, and joining data sources together is common, and comes with a set of challenges in optimizing the cost per HIT, the overall time to task completion, and the accuracy of MTurk results. We propose Qurk, a novel query system for managing these workflows, allowing crowd-powered processing of relational databases. We describe a number of query execution and optimization challenges, and discuss some potential solutions.

1. INTRODUCTION

Amazon's Mechanical Turk service (`https://www.mturk.com/mturk/welcome`) ("MTurk") allows users to post short tasks ("HITs") that other users ("turkers") can receive a small amount of money for completing. A HIT creator specifies how much he or she will pay for a completed task. Example HITs involve compiling some information from the web, labeling the subject of an image, or comparing two documents. More complicated tasks, such as ranking a set of ten items or completing a survey are also possible. MTurk is used widely to perform data analysis tasks which are either easier to express to humans than to computers, or for which there aren't yet effective artificial intelligence algorithms.

Task prices vary from a few cents ($.01-$.02/HIT is a common price) to several dollars for completing a survey. MTurk has around 100,000-300,000 HITs posted at any time (`http://mturk-tracker.com/general/`). Novel uses of MTurk include matching earthquake survivor pictures with missing persons in Haiti (`http://app.beextra.org/mission/show/missionid/605/mode/do`), authoring a picture book (`http://bjoern.org/projects/catbook/`), and embedding turkers as editors into a word processing system [2].

From the point of view of a database system, crowd-powered tasks can be seen as operators—similar to user-defined functions—that are invoked as a part of query processing. For example, given a database storing a table of images, a user might want to query for images of flowers, generating a HIT per image to have turkers perform the filter task. Several challenes arise in processing such a workflow. First, each HIT can take minutes to return, requiring an asynchronous query executor. Second, in addition to considering time, a crowdworker-aware optimizer must consider monetary cost and result accuracy. Finally, the selectivity of operators can not be predicted *a priori*, requiring an adaptive approach to query processing.

In this paper, we propose Qurk, a crowdworker-aware database system which addresses these challenges. Qurk can issue HITs that extract, order, filter, and join complex datatypes, such as images and large text blobs. While we describe Qurk here using the language of MTurk (turkers and HITs), and our initial prototype runs on MTurk, we aim for Qurk to be crowd-platform-independent. Future versions of Qurk will compile tasks for different kinds of crowds with different interfaces and incentive systems. Qurk is a new system in active development; we describe our vision for the system, propose a high level sketch of a UDF-like syntax for executing these HITs, and explore a number of questions that arise in such a system, including:

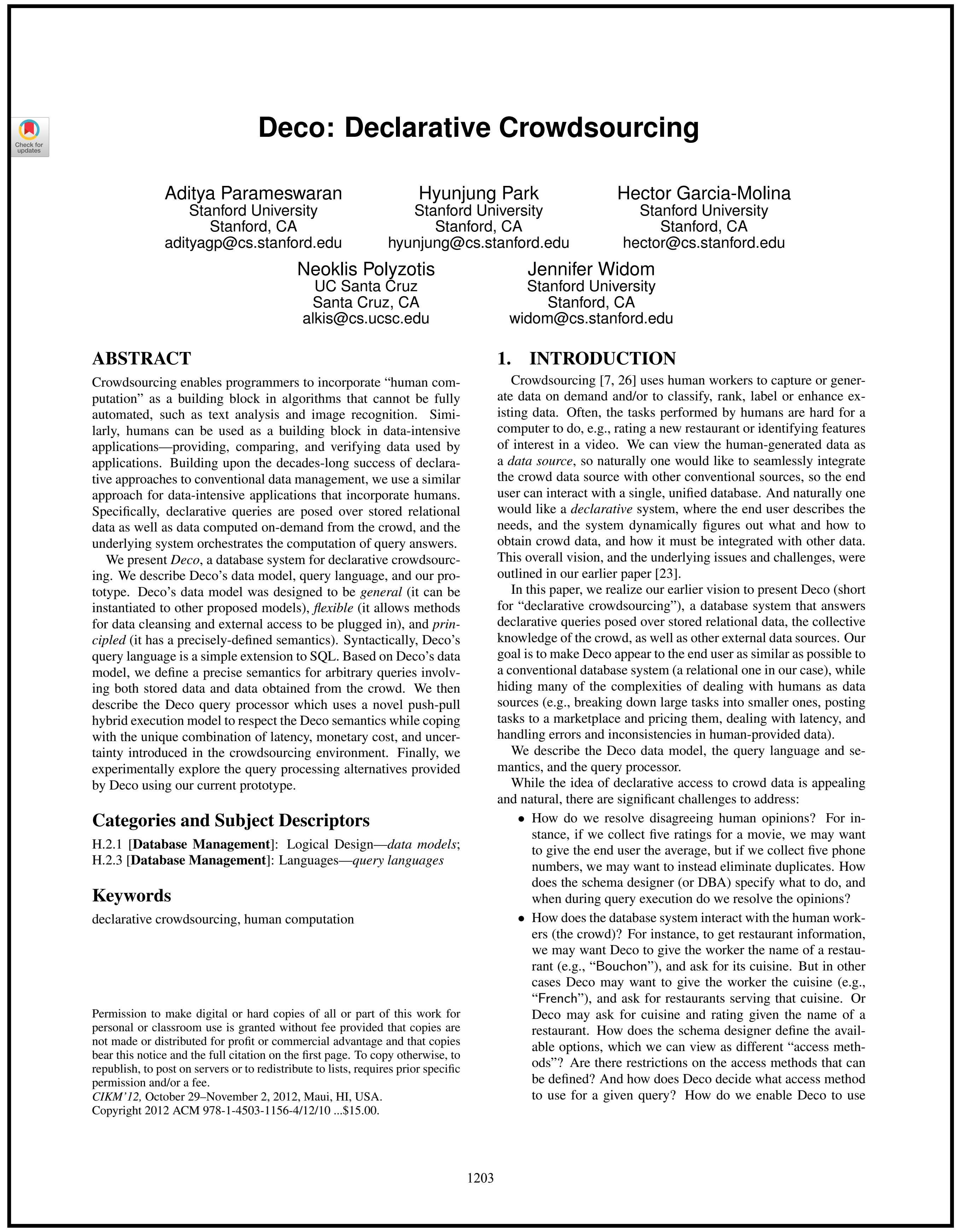
- What is the HIT equivalent of various relational operations? For example, an equijoin HIT might require humans to identify equal items, whereas a HIT-based sort can use human comparators.
- How many HITs should be generated for a given task? For example, when sorting, one can generate a HIT that asks users to score many items, or to simply compare two. How can the system optimize each HIT?
- Given that a large database could result in millions of HITs, how should the system choose which HITs to issue? Given that only a fraction of items in a large database can have HITs generated for them, what is the proper notion of query correctness?
- How much to charge for a HIT? Higher prices can lead to faster answers. Can the system adaptively optimize the price of HITs based on user-specified response time and budget constraints?
- Who is the right crowdworker for a task? MTurk provides paid workers, but an enterprise might prefer expert workers, and an operation with a tight budget might look for non-monetary incentives.

2. MOTIVATING EXAMPLES

Here is a list of tasks that Qurk should be able to run:

This article is published under a Creative Commons Attribution License (http://creativecommons.org/licenses/by/3.0/), which permits distribution and reproduction in any medium as well allowing derivative works, provided that you attribute the original work to the author(s) and CIDR 2011.
CIDR 2011

211



Deco: Declarative Crowdsourcing

Aditya Parameswaran
Stanford University
Stanford, CA
adityagp@cs.stanford.edu

Hyunjung Park
Stanford University
Stanford, CA
hyunjung@cs.stanford.edu

Hector Garcia-Molina
Stanford University
Stanford, CA
hector@cs.stanford.edu

Neoklis Polyzotis
UC Santa Cruz
Santa Cruz, CA
alkis@cs.ucsc.edu

Jennifer Widom
Stanford University
Stanford, CA
widom@cs.stanford.edu

ABSTRACT

Crowdsourcing enables programmers to incorporate "human computation" as a building block in algorithms that cannot be fully automated, such as text analysis and image recognition. Similarly, humans can be used as a building block in data-intensive applications—providing, comparing, and verifying data used by applications. Building upon the decades-long success of declarative approaches to conventional data management, we use a similar approach for data-intensive applications that incorporate humans. Specifically, declarative queries are posed over stored relational data as well as data computed on-demand from the crowd, and the underlying system orchestrates the computation of query answers.

We present *Deco*, a database system for declarative crowdsourcing. We describe Deco's data model, query language, and our prototype. Deco's data model was designed to be *general* (it can be instantiated to other proposed models), *flexible* (it allows methods for data cleansing and external access to be plugged in), and *principled* (it has a precisely-defined semantics). Syntactically, Deco's query language is a simple extension to SQL. Based on Deco's data model, we define a precise semantics for arbitrary queries involving both stored data and data obtained from the crowd. We then describe the Deco query processor which uses a novel push-pull hybrid execution model to respect the Deco semantics while coping with the unique combination of latency, monetary cost, and uncertainty introduced in the crowdsourcing environment. Finally, we experimentally explore the query processing alternatives provided by Deco using our current prototype.

Categories and Subject Descriptors

H.2.1 [**Database Management**]: Logical Design—*data models*;
H.2.3 [**Database Management**]: Languages—*query languages*

Keywords

declarative crowdsourcing, human computation

1. INTRODUCTION

Crowdsourcing [7, 26] uses human workers to capture or generate data on demand and/or to classify, rank, label or enhance existing data. Often, the tasks performed by humans are hard for a computer to do, e.g., rating a new restaurant or identifying features of interest in a video. We can view the human-generated data as a *data source*, so naturally one would like to seamlessly integrate the crowd data source with other conventional sources, so the end user can interact with a single, unified database. And naturally one would like a *declarative* system, where the end user describes the needs, and the system dynamically figures out what and how to obtain crowd data, and how it must be integrated with other data. This overall vision, and the underlying issues and challenges, were outlined in our earlier paper [23].

In this paper, we realize our earlier vision to present Deco (short for "declarative crowdsourcing"), a database system that answers declarative queries posed over stored relational data, the collective knowledge of the crowd, as well as other external data sources. Our goal is to make Deco appear to the end user as similar as possible to a conventional database system (a relational one in our case), while hiding many of the complexities of dealing with humans as data sources (e.g., breaking down large tasks into smaller ones, posting tasks to a marketplace and pricing them, dealing with latency, and handling errors and inconsistencies in human-provided data).

We describe the Deco data model, the query language and semantics, and the query processor.

While the idea of declarative access to crowd data is appealing and natural, there are significant challenges to address:

- How do we resolve disagreeing human opinions? For instance, if we collect five ratings for a movie, we may want to give the end user the average, but if we collect five phone numbers, we may want to instead eliminate duplicates. How does the schema designer (or DBA) specify what to do, and when during query execution do we resolve the opinions?
- How does the database system interact with the human workers (the crowd)? For instance, to get restaurant information, we may want Deco to give the worker the name of a restaurant (e.g., "Bouchon"), and ask for its cuisine. But in other cases Deco may want to give the worker the cuisine (e.g., "French"), and ask for restaurants serving that cuisine. Or Deco may ask for cuisine and rating given the name of a restaurant. How does the schema designer define the available options, which we can view as different "access methods"? Are there restrictions on the access methods that can be defined? And how does Deco decide what access method to use for a given query? How do we enable Deco to use

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
CIKM'12, October 29–November 2, 2012, Maui, HI, USA.
Copyright 2012 ACM 978-1-4503-1156-4/12/10 ...$15.00.

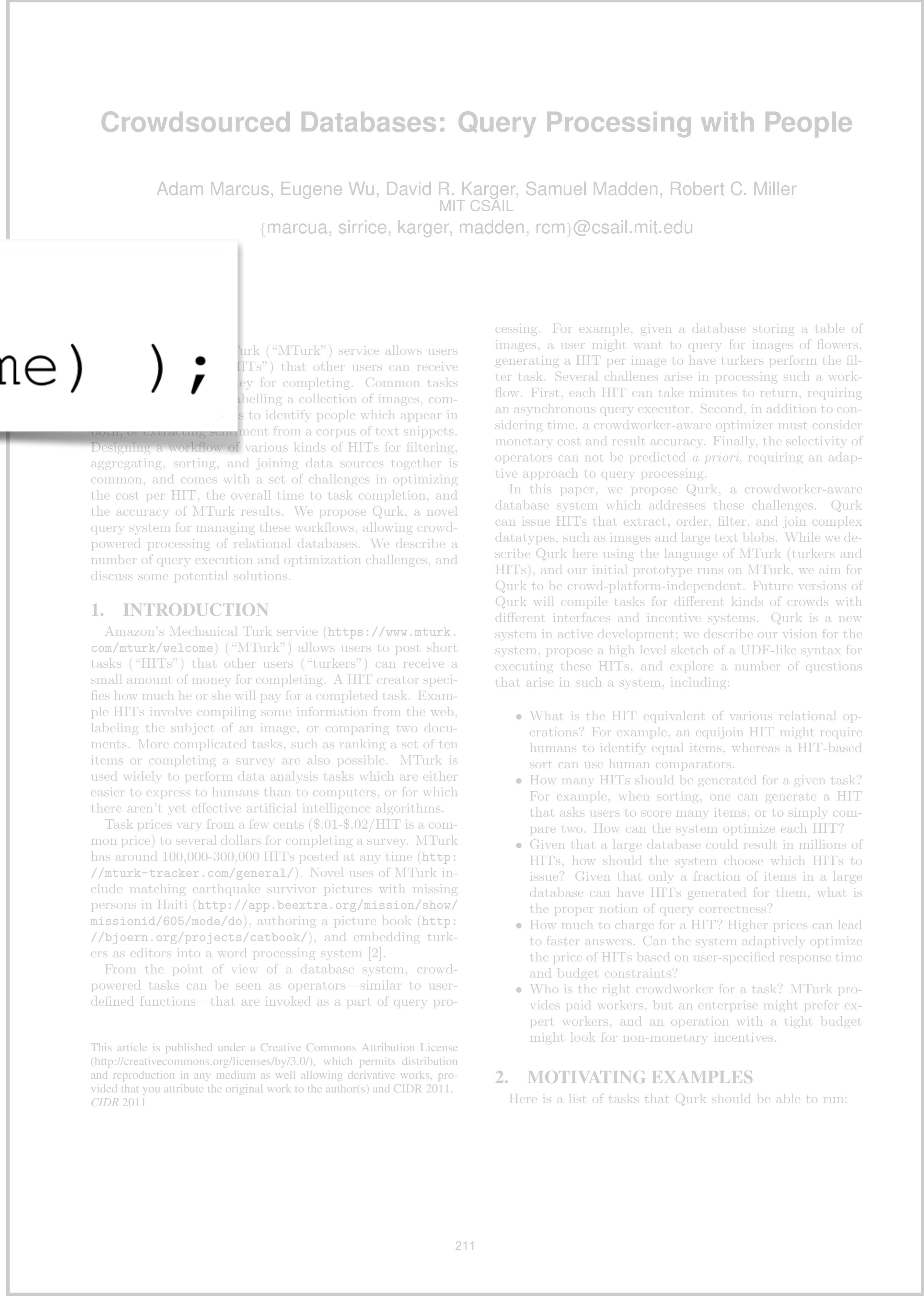
1203

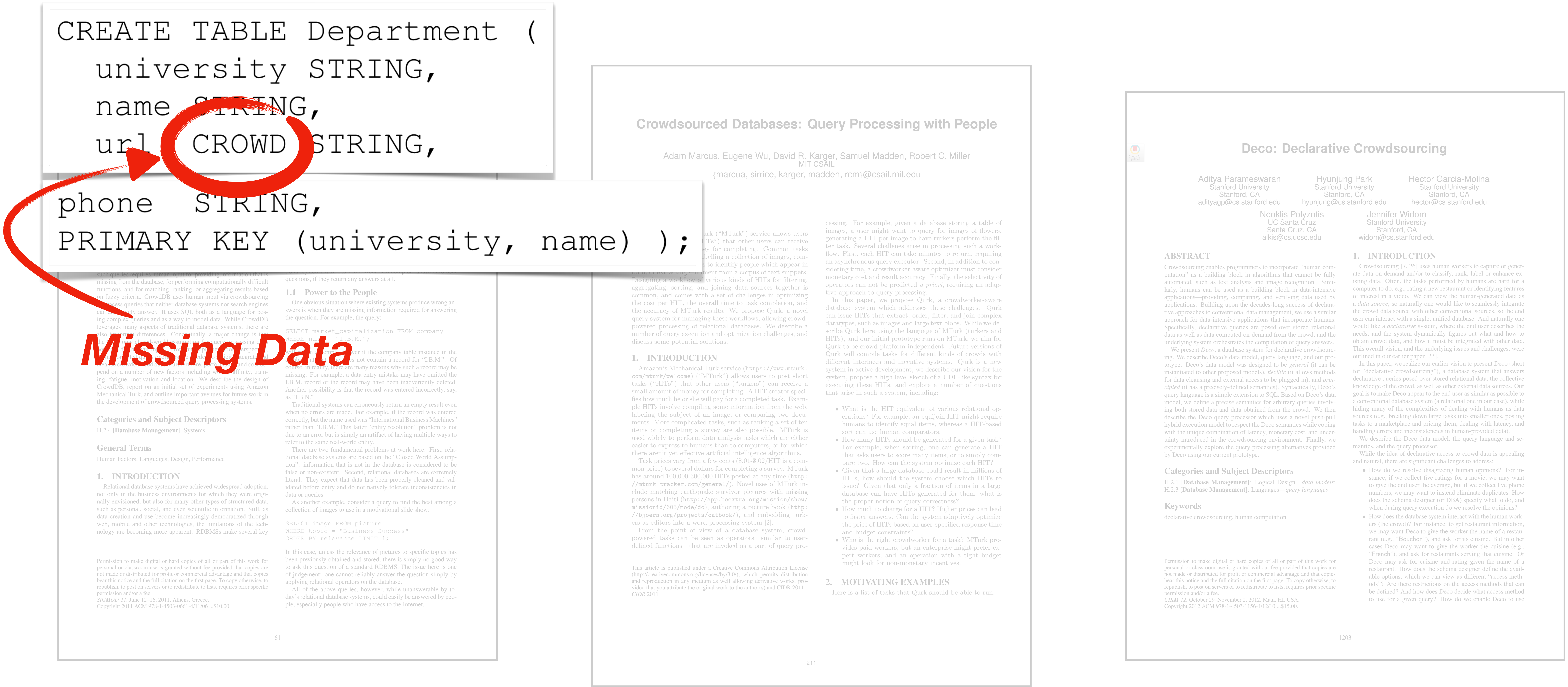
CrowdDB: Answering Queries with Crowdsourcing
Michael J. Franklin
AMPLab, UC Berkeley
franklin@cs.berkeley.edu
Donald Kossmann
Systems Group, ETH Zurich
donaldk@inf.ethz.ch
Tim Kraska
AMPLab, UC Berkeley
kraska@cs.berkeley.edu
Sukriti Ramesh
Systems Group, ETH Zurich
ramess@student.ethz.ch
Reynold Xin
AMPLab, UC Berkeley
rxin@cs.berkeley.edu
ABSTRACT

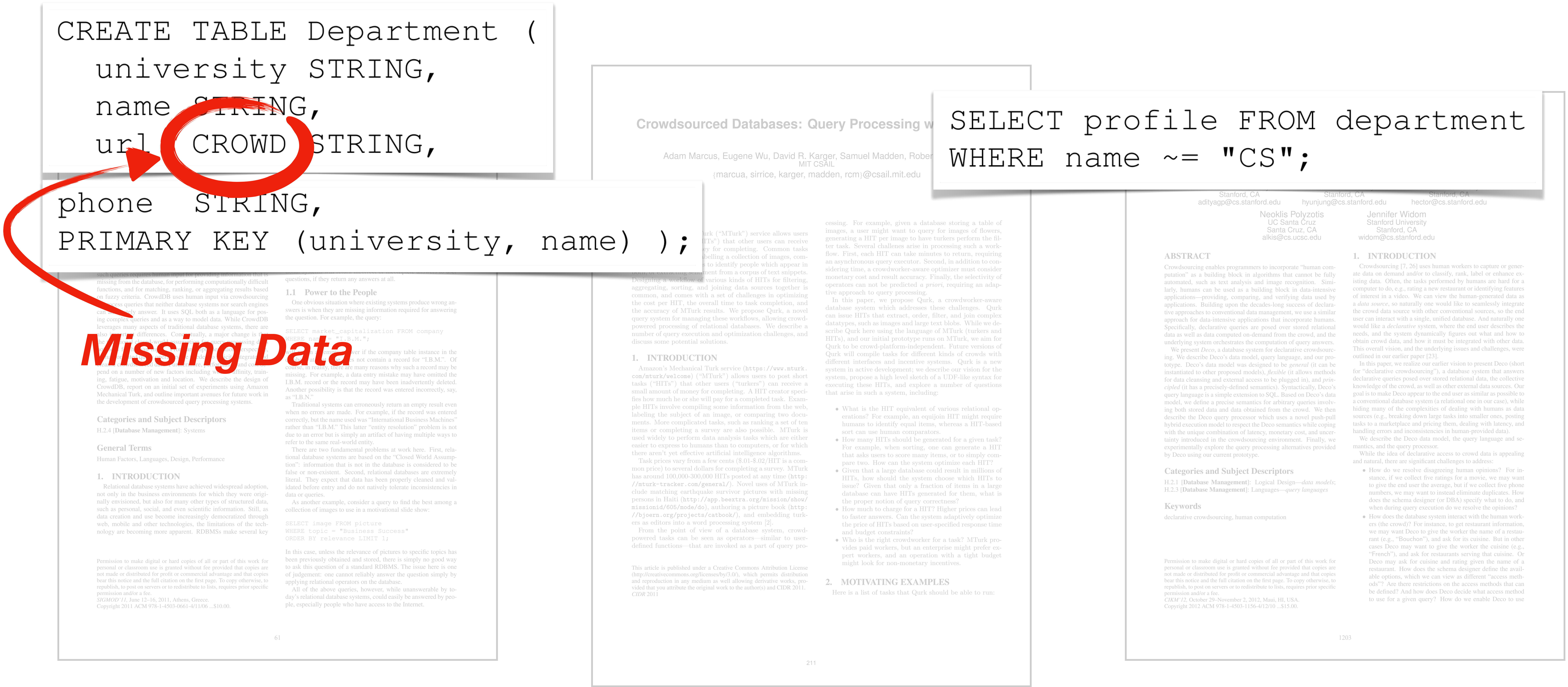
Crowdsourced Databases: Query Processing with People
Adam Marcus, Eugene Wu, David R. Karger, Samuel Madden, Robert C. Miller
MIT CSAIL
{marcua, sirrice, karger, madden, rcm}@csail.mit.edu
ABSTRACT

Deco: Declarative Crowdsourcing
Aditya Parameswaran
Stanford University
Stanford, CA
adityagp@cs.stanford.edu
Hyunjung Park
Stanford University
Stanford, CA
hyunjung@cs.stanford.edu
Hector Garcia-Molina
Stanford University
Stanford, CA
hector@cs.stanford.edu
Neoklis Polyzotis
UC Santa Cruz
Santa Cruz, CA
alkis@cs.ucsc.edu
Jennifer Widom
Stanford University
Stanford, CA
widom@cs.stanford.edu
ABSTRACT
1. INTRODUCTION

```
CREATE TABLE Department (
  university STRING,
  name STRING,
  url  CROWD STRING,
phone  STRING,
PRIMARY KEY (university, name) );
```




CREATE TABLE Department (
university STRING,
name STRING,
url CROWD STRING,
phone STRING,
PRIMARY KEY (university, name));
Missing Data
Crowdsourced Databases: Query Processing with People
Adam Marcus, Eugene Wu, David R. Karger, Samuel Madden, Robert C. Miller
MIT CSAIL
{marcua, sirrice, karger, madden, rcm}@csail.mit.edu
Deco: Declarative Crowdsourcing
Aditya Parameswaran
Hyunjung Park
Hector Garcia-Molina
Neoklis Polyzotis
Jennifer Widom


CREATE TABLE Department (
university STRING,
name STRING,
url CROWD STRING,
phone STRING,
PRIMARY KEY (university, name));
Missing Data
Crowdsourced Databases: Query Processing w
Adam Marcus, Eugene Wu, David R. Karger, Samuel Madden, Rober
MIT CSAIL
{marcua, sirrice, karger, madden, rcm}@csail.mit.edu
SELECT profile FROM department
WHERE name ~= "CS";

Comparison Using Commonsense Knowledge

```
CREATE TABLE Department (
   university STRING,
   name STRING,
   url CROWD STRING,
phone  STRING,
PRIMARY KEY (university, name) );
```

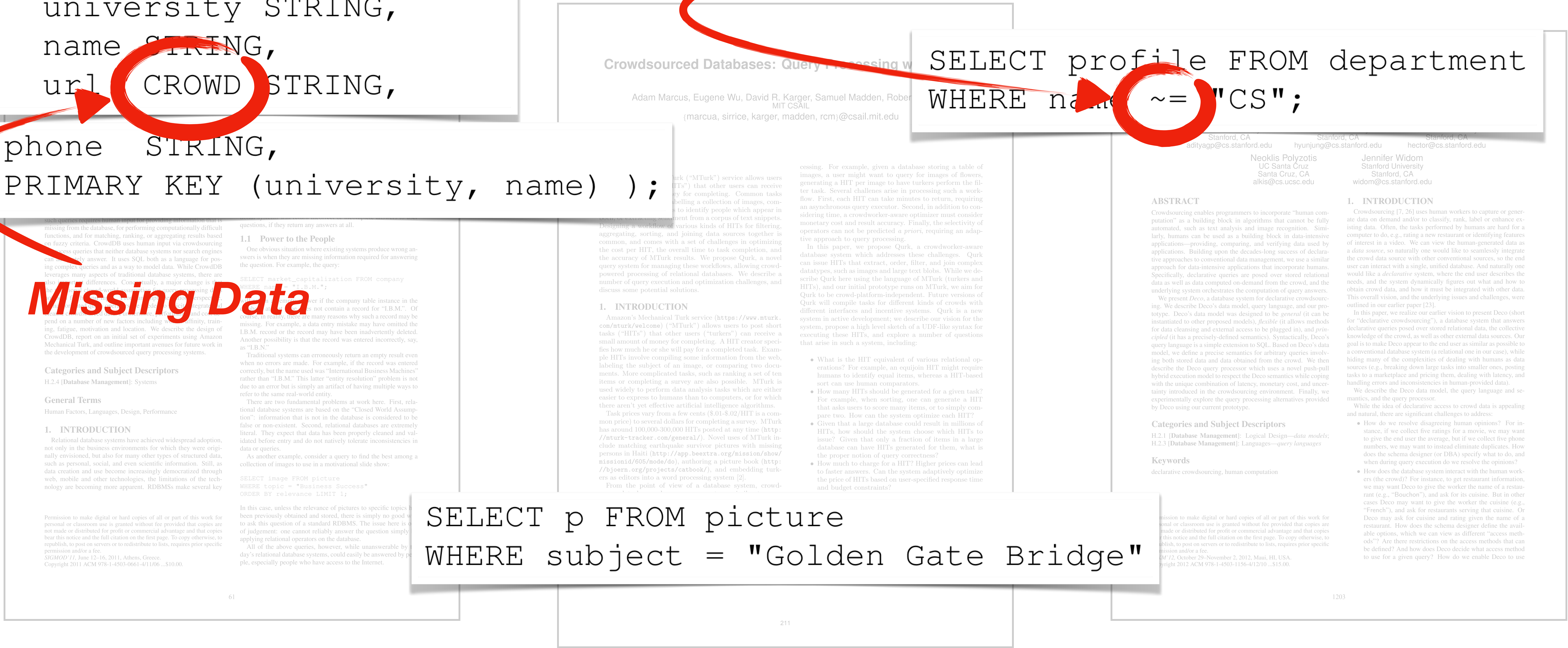
Missing Data

```
SELECT profile FROM department
WHERE name ~= "CS";
```

Comparison Using Commonsense Knowledge

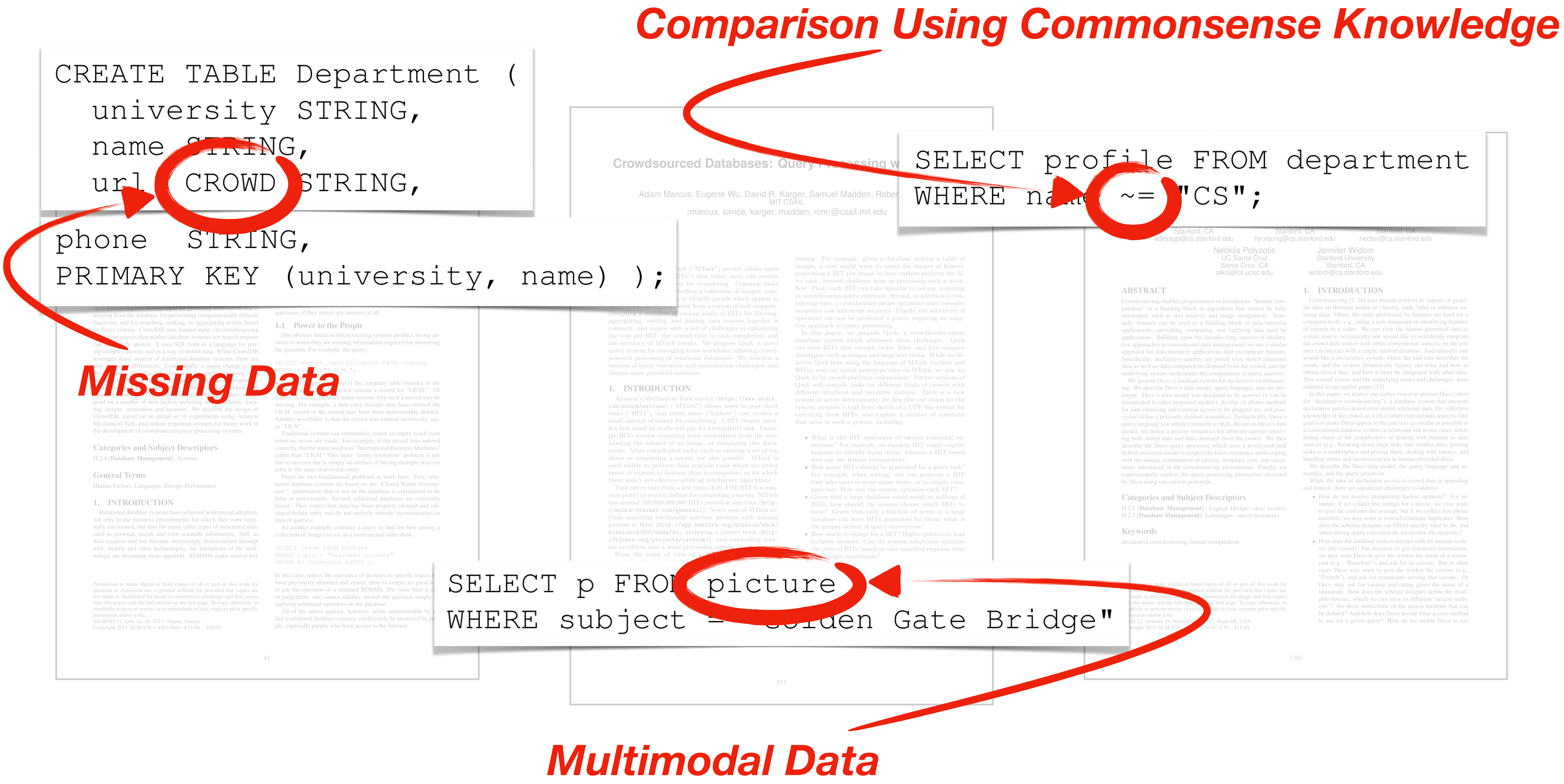
```
CREATE TABLE Department (
  university STRING,
  name STRING,
  url CROWD STRING,
phone  STRING,
PRIMARY KEY (university, name) );
```

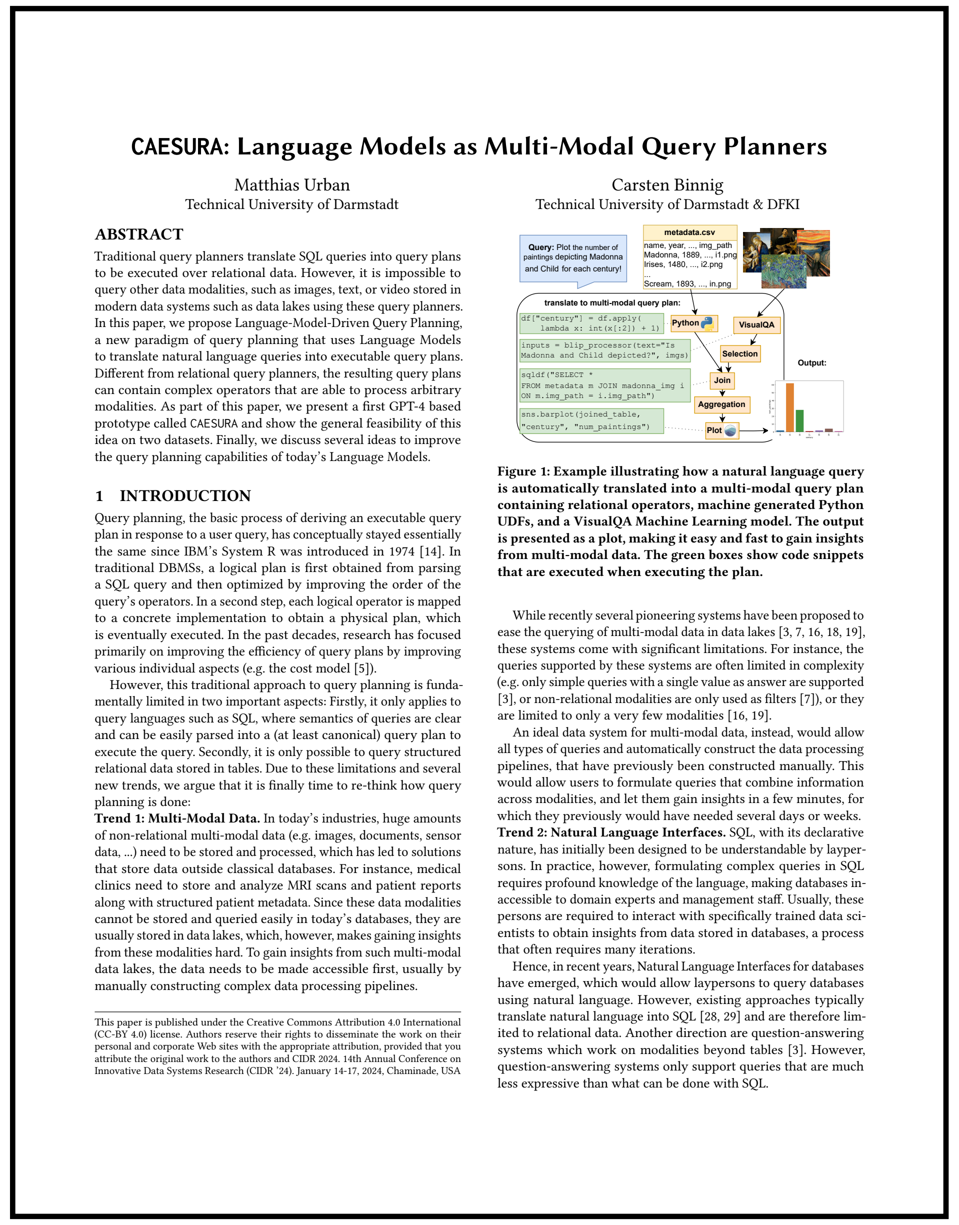
Missing Data



```
SELECT profile FROM department
WHERE name ~= "CS";
```

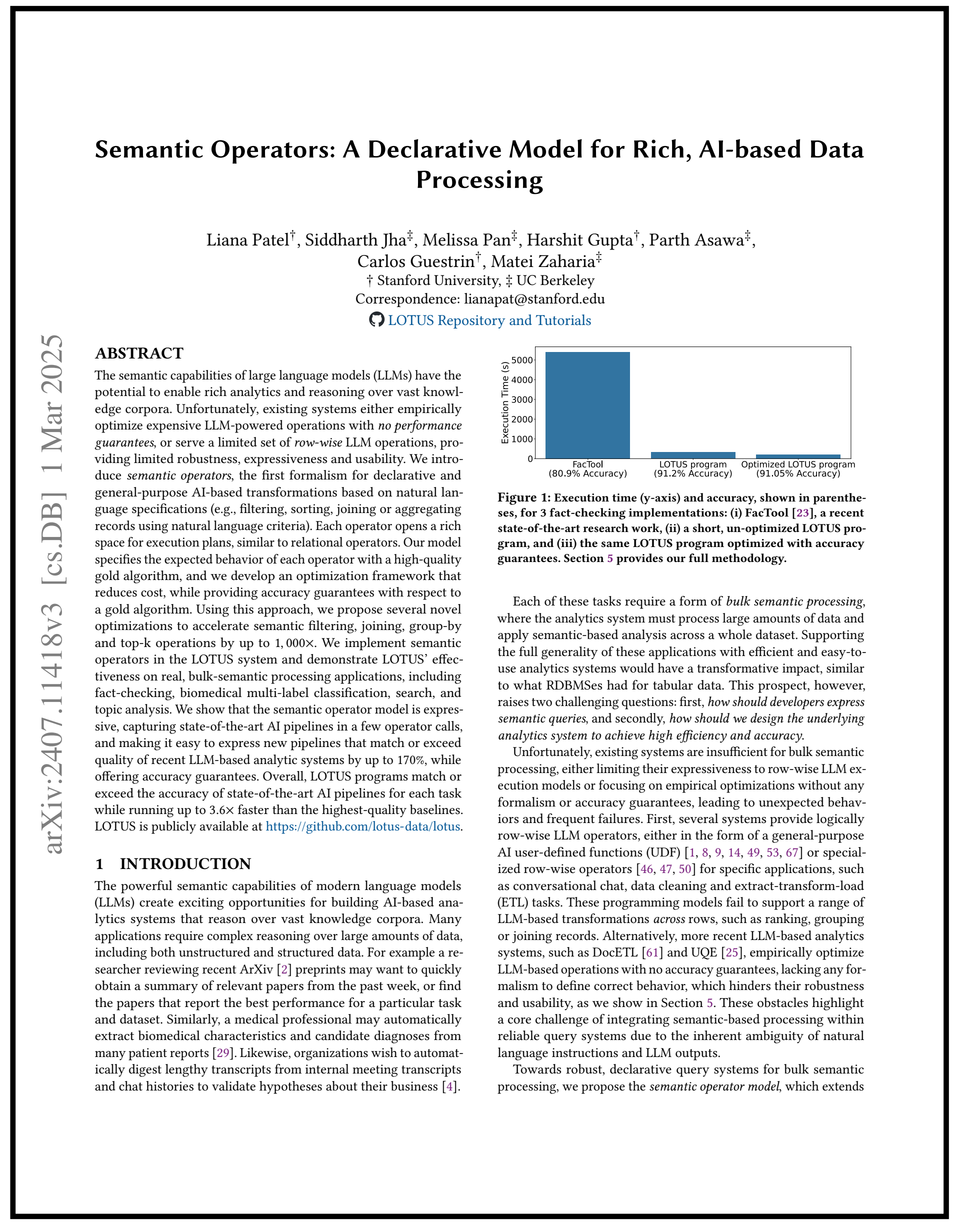
```
SELECT p FROM picture
WHERE subject = "Golden Gate Bridge"
```

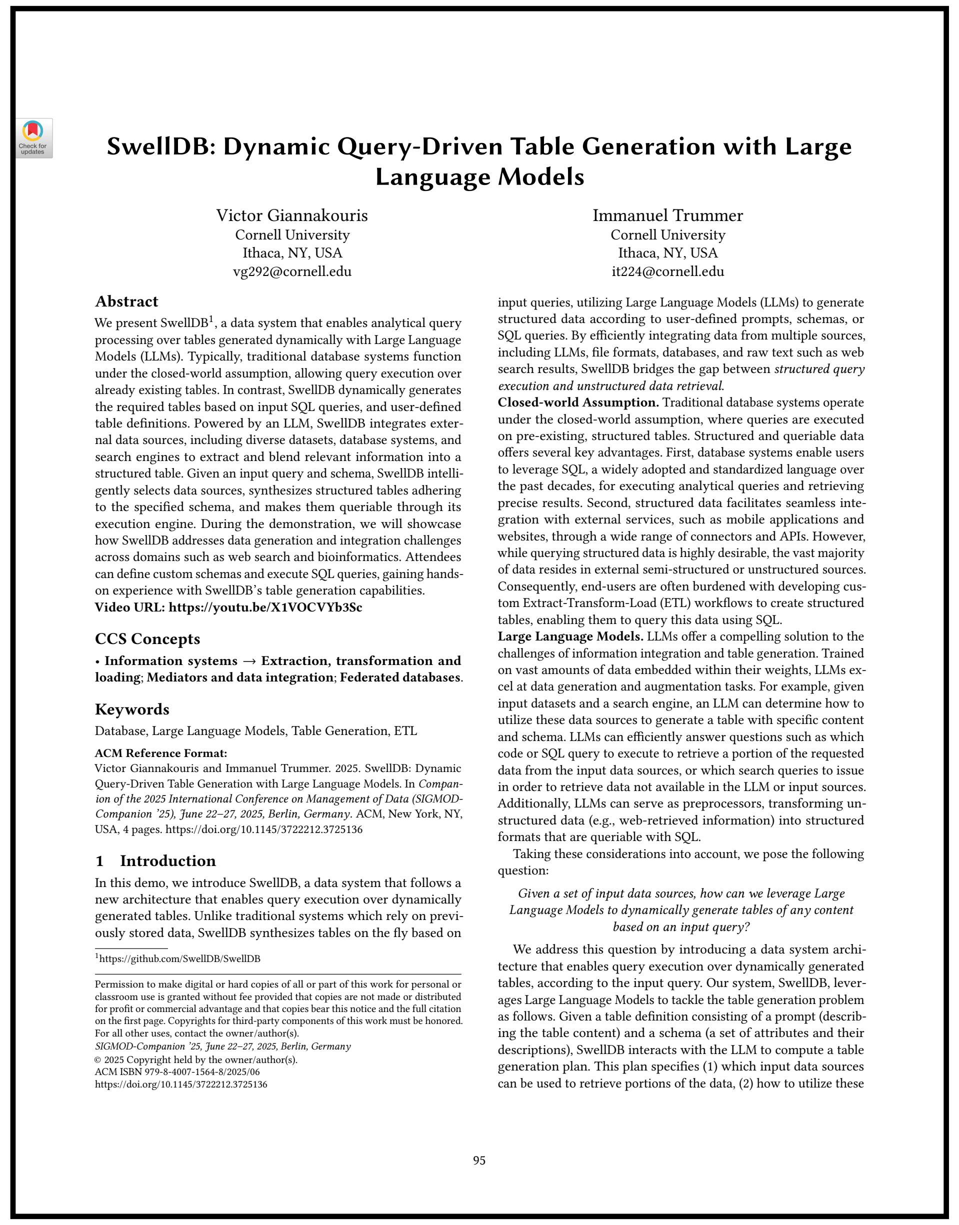

Comparison Using Commonsense Knowledge
CREATE TABLE Department (
university STRING,
name STRING,
url CROWD STRING,
phone STRING,
PRIMARY KEY (university, name));
Crowdsourced Databases: Query Processing w
SELECT profile FROM department
WHERE name ~= "CS";
Missing Data
SELECT p FROM picture
WHERE subject = "Golden Gate Bridge"
Multimodal Data


CAESURA: Language Models as Multi-Modal Query Planners
Matthias Urban
Carsten Binnig


Palimpzest: Optimizing AI-Powered Analytics with Declarative Query Processing


ThalamusDB: Approximate Query Processing on Multi-Modal Data
SAEHAN JO, Cornell University, USA
IMMANUEL TRUMMER, Cornell University, USA


Semantic Operators: A Declarative Model for Rich, AI-based Data Processing


SwellDB: Dynamic Query-Driven Table Generation with Large Language Models
Victor Giannakouris
Immanuel Trummer

CAESURA: Language Models as Multi-Modal Query Planners

Palimpzest: Optimizing AI-Powered Analytics with Declarative Query Processing

ThalamusDB: Approximate Query Processing on Multi-Modal Data

Semantic Operators: A Declarative Model for Rich, AI-based Data Processing

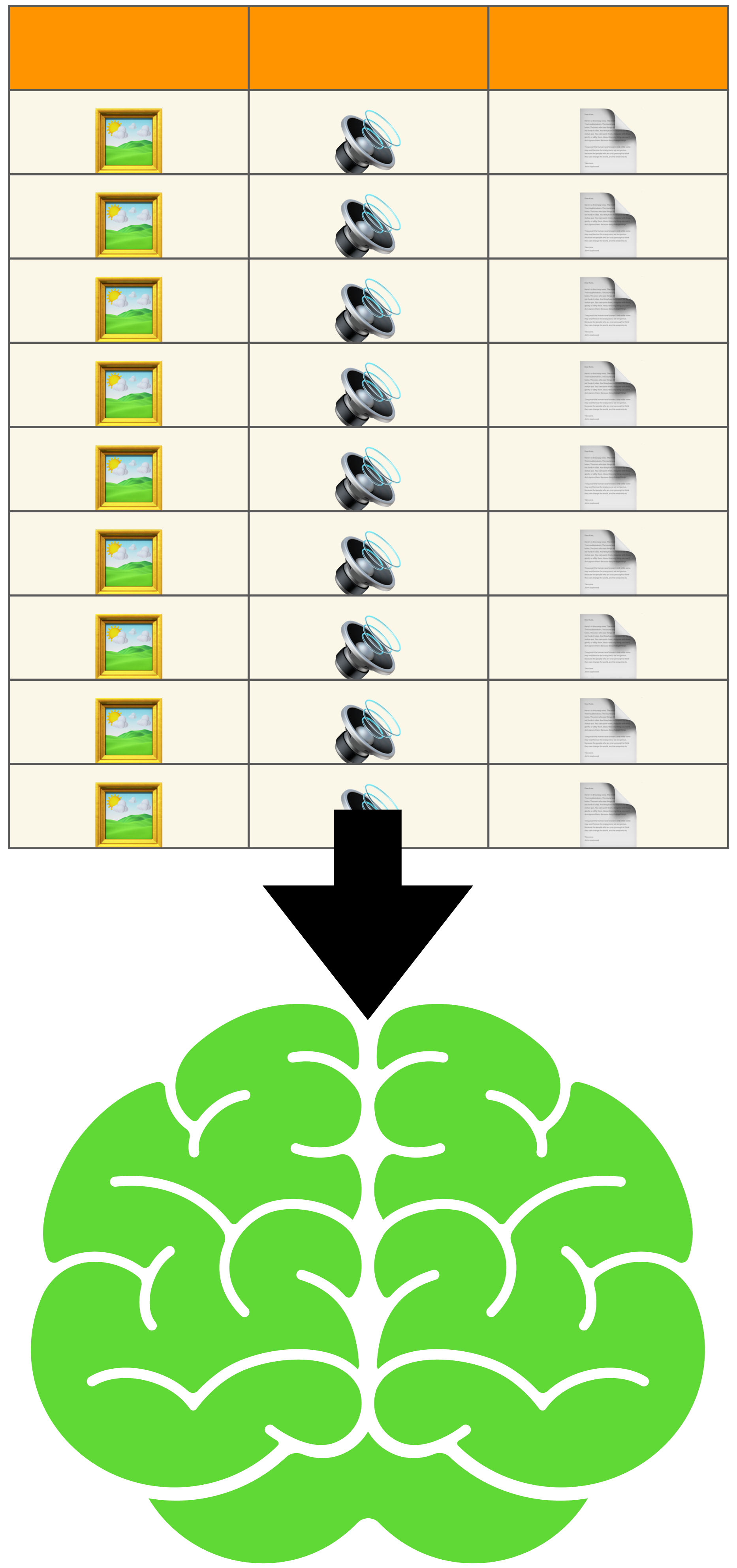
SwellDB: Dynamic Query-Driven Table Generation with Large Language Models

Efficiency?

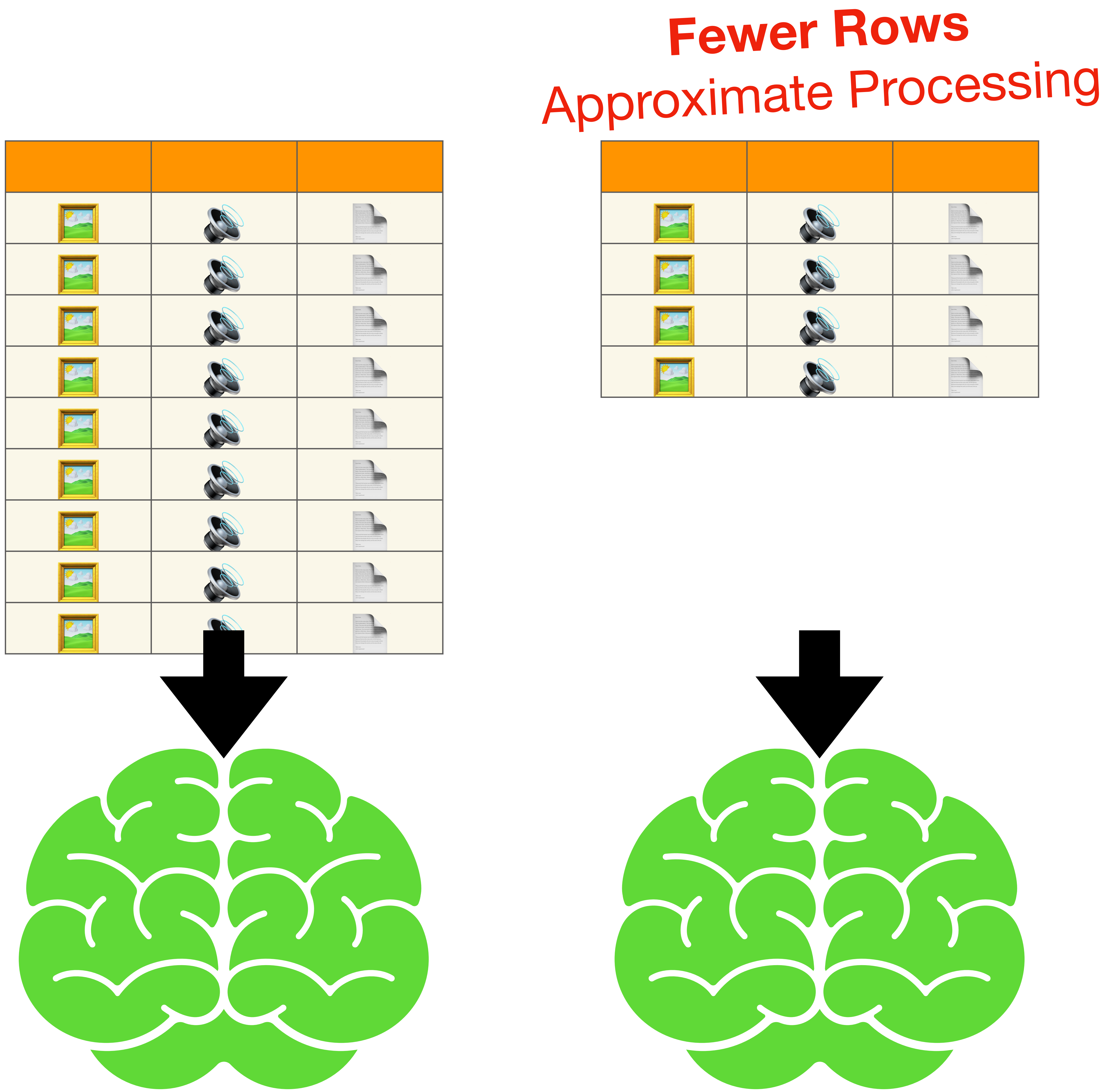
Lowering LLM Costs

🖼️	🔊	📄
🖼️	🔊	📄
🖼️	🔊	📄
🖼️	🔊	📄
🖼️	🔊	📄
🖼️	🔊	📄
🖼️	🔊	📄
🖼️	🔊	📄
🖼️	🔊	📄

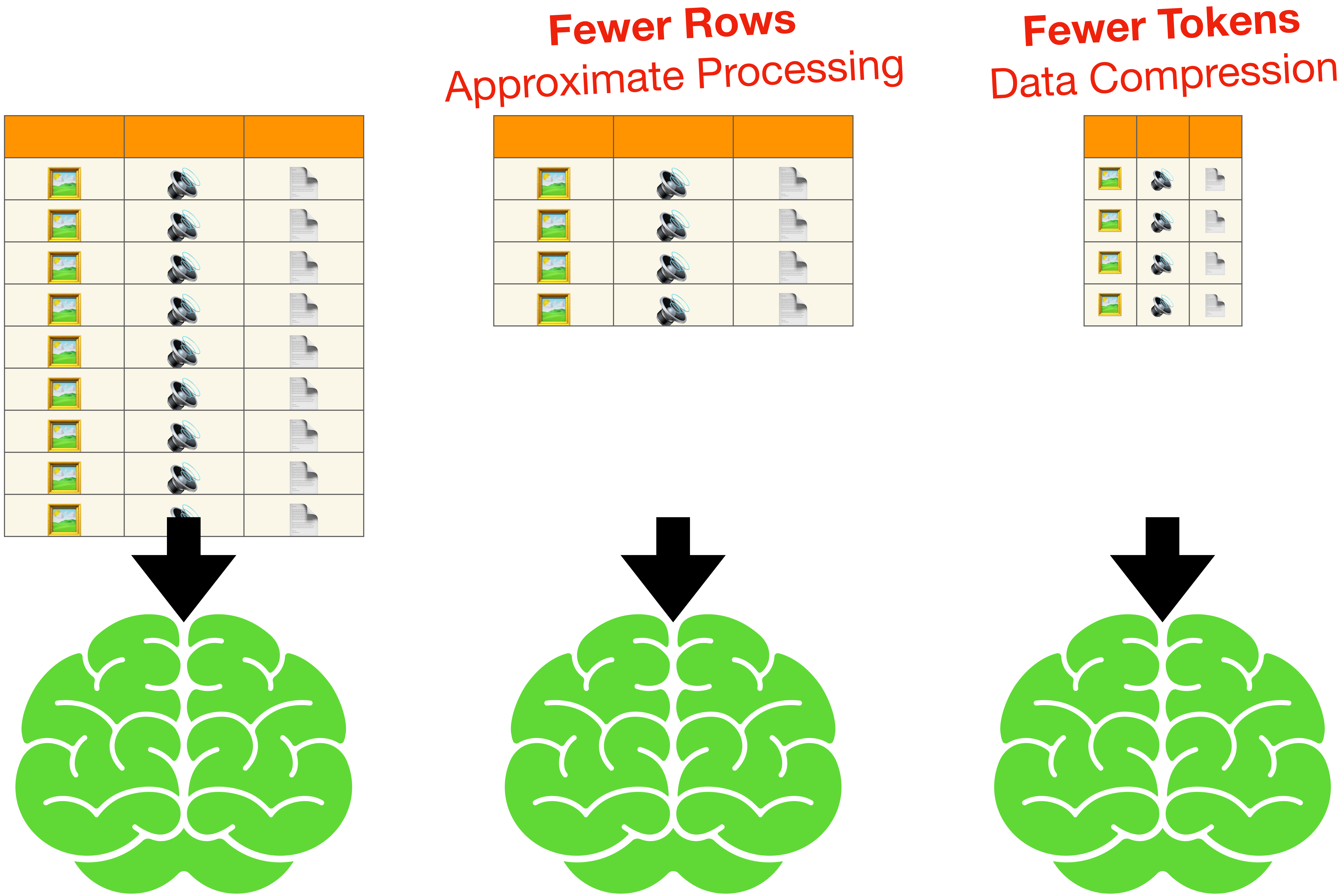
Lowering LLM Costs



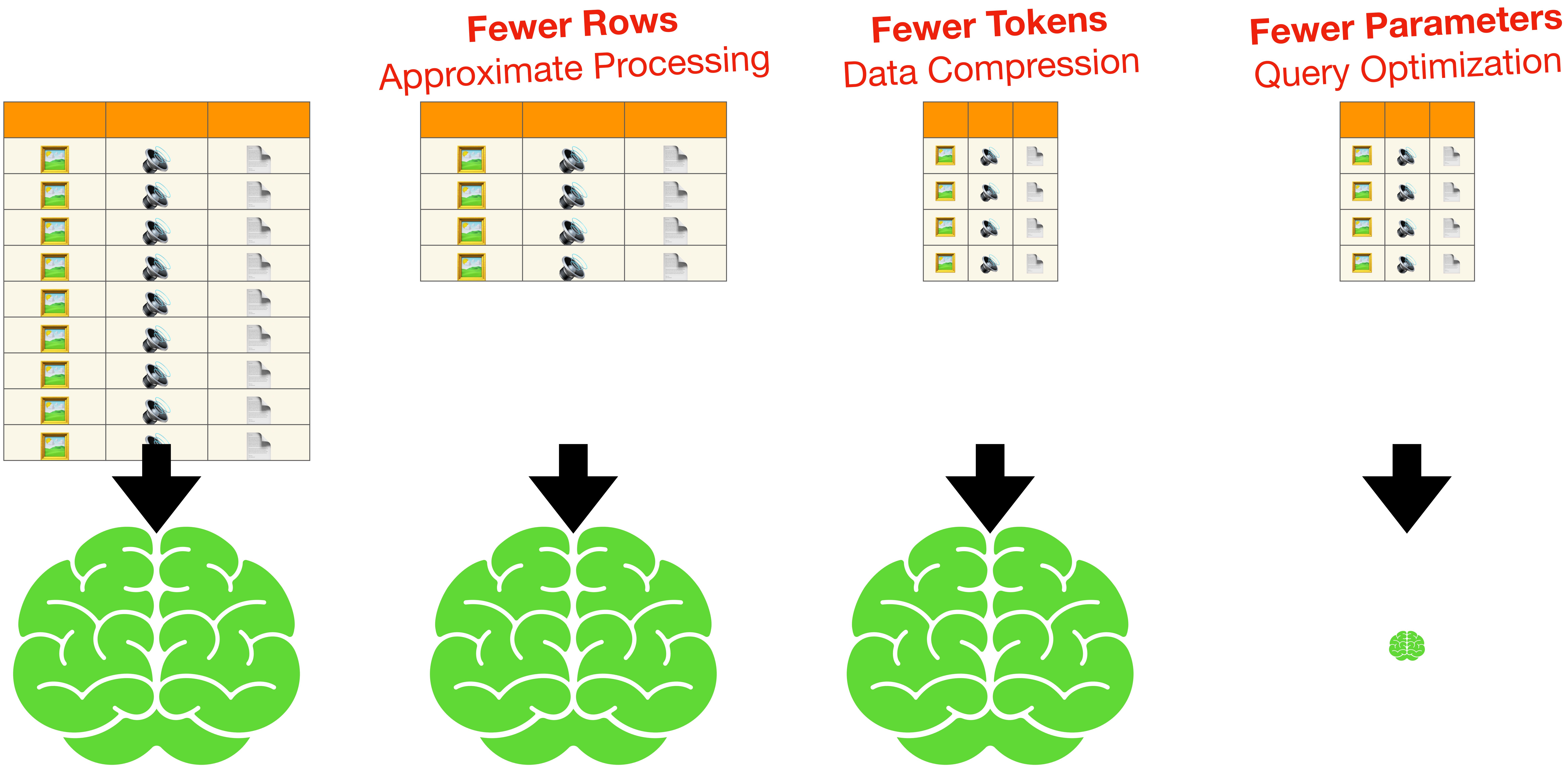
Lowering LLM Costs



Lowering LLM Costs



Lowering LLM Costs



Summary

- Dramatic **expansion of scope** in data processing
  - Data location
  - Data types
  - DB Operators
- Key challenge: **scale it up**!
  - Approximate processing
  - Specialized operators
  - Query optimization

?